On Nominal Delay Minimization
in LUT-Based FPGA Technology Mapping

Jason Cong and *Yuzheng Ding*

Department of Computer Science
University of California, Los Angeles, CA 90024

Abstract

In this report, we study the LUT-based FPGA technology mapping problem for delay minimization under the nominal delay model, which assumes that the interconnect delay of a net is proportional to the fanout size of the net. First, we prove several complexity results on LUT mapping under the nominal delay model. We prove that the delay-optimal K-LUT mapping problem under the nominal delay model is NP-hard when $K \geq 3$, and remain NP-hard for duplication-free mapping and tree-based mapping for $K \geq 5$. Also, we show that for $K = 2$ the delay-optimal duplication-free mapping or tree-based mapping under the nominal delay model can be solved in polynomial time. Next, we develop a heuristic LUT mapping algorithm for nominal delay minimization. Experimental results have shown that our heuristic algorithm can produce mapping solutions of smaller delay compared with the solutions of depth-optimal mapping algorithm under the unit delay model.

Area of Interest: (1) Combinational Logic Synthesis.

1. Introduction

Lookup-table (LUT) based FPGA [11, 21] is a popular architecture in which the basic programmable logic block is a K-input lookup table (K-LUT), built in SRAM, which can implement any Boolean function of up to K variables. The technology mapping problem in LUT-based FPGA designs is to transform a general Boolean network into a functionally equivalent network of K-LUTs by computing a (not necessarily disjoint) K-LUT covering of the network.

Extensive study has been done on the mapping algorithms for LUT-based FPGAs in recent years. Researchers have focused on area minimization [7, 8, 12, 13, 15, 16, 20], delay minimization [2, 4, 6, 9, 14, 17], trade-off between area and delay [5], and routability optimization [1, 19]. Efforts have been made both on the development of effective and efficient mapping algorithms, and on the study of the complexity of the mapping problems. It has been shown that if the network is a tree, or if we use *tree-based mapping* (i.e. by decomposing a general network into trees and mapping each tree separately), both the area minimization problem and the depth minimization problem can be solved optimally in polynomial time [7, 8, 9]. For general K-bounded Boolean networks, it is shown that the depth minimum mapping problem can be solved optimally in strong polynomial time [4], while the area minimum mapping problem is NP-hard for $K \geq 5$ [7]. If we allow only *duplication-free mapping* (i.e. do not allow node duplication during mapping), it has been shown that both the depth minimum mapping problem and the area minimum mapping problem can be solved optimally in polynomial time for any fixed K [5]. These results are summarized in Table 1. For the NP-hard area minimization problem, efficient heuristics have also been developed.

<table>
<tr><td></td><td>tree-based mapping</td><td>dup-free mapping</td><td colspan="2">general mapping</td></tr>
<tr><td rowspan="2">area minimization</td><td rowspan="2">$O(\min\{nK,\ n\log n\})$
[7]</td><td rowspan="2">$O(n^{K+1})$
[5]</td><td>$2 \leq K < 5$</td><td>$K \geq 5$</td></tr>
<tr><td>open</td><td>NP-hard [7]</td></tr>
<tr><td>depth minimization</td><td>$O(\min\{nK,\ n\log n\})$
[7]</td><td>$O(Kmn)$
[4]</td><td colspan="2">$O(Kmn)$
[4]</td></tr>
</table>

Table 1 Summary of previous complexity results on LUT mapping
(Mapping K-bounded network with n nodes and m edges into K-LUTs)

Delay minimization has been an important optimization objective in FPGA mapping because the speed of FPGA designs is usually slower than that of the gate array or standard cell designs due to the extra delay introduced by the programmable interconnects on FPGA chips. Most previous mapping algorithms for delay minimization use the depth of the mapping solution as the measurement of delay, i.e. based on the *unit delay model*, which assumes uniform delay at every logic level. As shown in [4], the depth minimization problem can be solved optimally in polynomial time by efficient network flow computation. However, the assumption made by the unit delay model is usually over-simplified. In LUT-based FPGA designs, although the delay of each LUT is a constant, the interconnect delay of each net may vary considerably. Moreover, interconnect delay contributes a significant portion to the total delay. This portion of delay cannot be accurately estimated by the unit delay model. Therefore, delay models should allow variable delay values for different nets.

Experiments in [18] have shown that the interconnect delay in an FPGA chip is closely related to the number of fanouts of a signal. This is intuitively true because an FPGA chip has fixed routing resources, a net with larger fanouts must spread over a larger area and use up more routing resources, and thus has larger delay. Although the actual delay of a net depends on the final routing result, it is reasonable to estimate the net delay based on the number of fanouts of the net in the delay model for mapping.

In this report we study the LUT-based FPGA technology mapping problem for delay minimization under the *nominal delay model*, which assumes the net delay proportional to the net fanout size. Under this model we show that the delay-optimal mapping problem for K-LUTs becomes NP-hard for $K \geq 3$, and remains NP-hard for $K \geq 5$ even for duplication-free mapping and tree-based mapping, which is a surprising result, considering that tree-based mapping for area minimization can be solved efficiently. We also show that when $K=2$, the duplication-free mapping and tree-based mapping problems for nominal delay minimization can be solved in polynomial time. These complexity results are summarized in Table 2.

	$k=2$	$K=3$, $K=4$	$K \geq 5$
tree-based mapping	$O(mn)$	open	NP-hard
dup-free mapping	$O(mn)$	open	NP-hard
general mapping	open	NP-hard	

Table 2 Summary of complexity results on LUT mapping shown in this report
(Delay-optimal mapping of K-bounded network with n nodes and m edges into K-LUTs under nominal delay model)

In general, mapping based on the nominal delay model can produce better solutions compared with unit delay model based mapping. Using a simple heuristic presented in this report, we are able to achieve consistent improvement over the depth-optimal mapping algorithm in terms of the actual delay in the final FPGA designs. To our knowledge, this is the first report which presents in-depth study of the nominal delay minimization problem in LUT mapping.

The remainder of this report is organized as follows. Section 2 presents the problem formulation and introduces the nominal delay model. In Section 3, we present the complexity results of the delay minimization problems in LUT-based FPGA technology mapping. In Section 4, we present mapping algorithms, including a polynomial time optimal algorithm for a special case and a heuristic for the general case. Our experimental results are reported in Section 5. Section 6 concludes the report.

2. Problem Formulation & Nominal Delay Model

2.1. Problem Formulation

A combinational Boolean network is represented as a directed acyclic graph (DAG) where each node represents a logic gate, and a directed edge $(i,\ j)$ exists if the output of gate i is an input of gate j. A primary input (PI) node has no incoming edge and a primary output (PO) node has no outgoing edge. The set of fanins of gate v is denoted $input(v)$, and the set of *distinct* nodes which supply inputs to the gates in subnetwork H is denoted $input(H)$. Similarly, the set of fanouts of v is denoted $output(v)$, and the set of distinct fanouts of a subnetwork H is denoted $output(H)$. The *level* (or *depth*) of a node v is the number of edges on the longest path from any PI node to v. The *depth* of a network is the largest node level in the network. A Boolean network is *K-bounded* if $|\ input(v)| \le K$ for every node v. In the remaining of the report we assume that the networks to be mapped are K-bounded unless otherwise specified[1].

For a node v in the network, a *cone of* v, denoted C_v, is a subgraph of logic gates (excluding PIs) consisting of v and its predecessors such that any path connecting a node in C_v and v lies entirely in C_v. The *root* of C_v is v. The *fanin cone* of node v, denoted N_v, consists of v and all the predecessors of v. A *fanout-free cone (FFC) at* v, denoted FFC_v, is a cone of v such that for any node $u \neq v$ in FFC_v, $output(u) \subseteq FFC_v$. A *K-feasible cone of* v is a cone C_v such that $|input(C_v)| \le K$. A *cut* in a fanin cone N_v of node v is a bipartition $(X,\ \overline{X})$ of N_v such that $\overline{X}$ is a cone of v, and $w \in X$ for every PI node $w \in N_v$. If $\overline{X}$ is a K-feasible cone, the cut is called a *K-feasible cut*.

A K-LUT LUT_v that implements node v *covers* a K-feasible FFC C_v of v. If C_v is not fanout free, the non-root nodes in C_v that have fanouts outside of C_v must be duplicated in order to cover C_v by a K-LUT. Given a K-bounded network, the *technology mapping problem* for K-LUT based FPGA designs is to cover the network with K-feasible FFCs, possibly with node duplications. If node duplication is not allowed, it is *duplication-free mapping*. If the network is first decomposed into trees, and each tree is then mapped separately, it is *tree-based* mapping.

---

[1] Some existing mapping algorithms can handle unbounded networks by incorporating node decomposition in the mapping procedure [8, 9].

2.2. Nominal Delay Model & Difficulty of Nominal Delay Minimization in LUT Mapping

Under the *nominal delay model*, the delay of a K-LUT LUT_v and its fanout net is estimated as

$$D(LUT_v) = d_L + |\ output(v)\ | * d_N, \tag{1}$$

where d_L is the delay of a K-LUT, which is a constant for a given technology, and $d_N > 0$ is a constant representing the additional delay due to adding a fanout branch to the net (which is determined by the technology, the placement/routing tools, and the style of the design, etc.)[2]. In practice, d_N is usually not a constant. Therefore, the nominal delay is still a simplified approximation of the real delay. Nevertheless, the nominal delay model is an improvement over the simple unit delay model in terms of estimating the net delays, and still simple enough to be considered during the technology mapping stage.

The PI and PO nodes usually have different delays from the K-LUTs, but since there will be exactly one PI and one PO along each critical path, their delay values will not affect the mapping solution. The delays of the fanout nets of the PI nodes will also be estimated using the nominal delay model.

The nominal delay minimization problem is much more difficult than the depth minimization problem in LUT mapping due to the following reason. Under the unit delay model, for any node v, the minimum depth of a K-LUT that implements v only depends on the depth-optimal mapping of the fanin cone N_v of v. This allows the depth-optimal solution to be computed using dynamic programming approach. Under the nominal delay model, however, the minimum delay of a K-LUT that implements v may also depend on the mapping of nodes outside of N_v. Figure 1 illustrates this scenario. Let u be a predecessor of v which has fanouts x and y outside of N_v. If both nodes x and y are packed into one LUT, the nominal delay of node u decreases, since the fanout size of u is decreased by one (assuming that u will be implemented by a K-LUT). However, if x or y is duplicated, the nominal delay of u will increase. Therefore, delay-optimal mapping of v clearly depends on the mapping of the nodes x and y outside of N_v. This is the inherent difficulty of nominal delay minimization in LUT mapping, which leads to the NP-hardness results in the next section.

---

[2] Note that if we set d_N=0, the nominal delay model becomes the unit delay model.

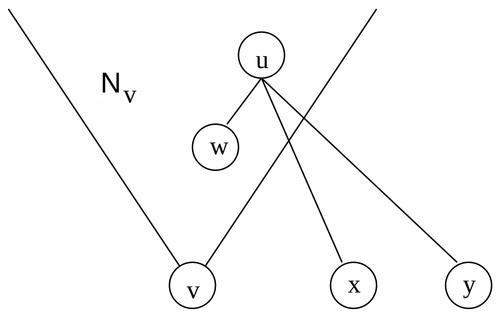

Figure 1 Complication of mapping under nominal delay model.

3. NP-Hardness Results on Nominal Delay Minimization in LUT Mapping

Our proof of the NP-hardness of the delay-optimal LUT mapping problem will be based on polynomial-time transformations from the **RSAT** to the decision version of the delay-optimal LUT mapping problem under the nominal delay model. The **RSAT** problem is a restricted case of the well-known NP-complete problem **SAT**, and remains NP-complete. We start with the definition of the **RSAT** problem.

Problem: Restricted Satisfiability (**RSAT**)
Instance: A set of n Boolean variables $X = \{x_1, x_2, ..., x_n\}$ and a collection of m clauses $C = \{C_1, C_2, ..., C_m\}$, where (1) each clause is the disjunction (OR) of 2 or 3 literals of the variables, (2) each clause contains at most one of x_i and $\overline{x}_i$ for any variable x_i, (3) both x_i and $\overline{x}_i$ of any variable x_i appear in some of the clauses, and (4) for any variable x_i, the number of clauses containing either x_i or $\overline{x}_i$ is no more than 3.
Question: Is there a truth assignment of the variables in X such that $C_j = 1$ for $1 \le j \le m$?

Lemma 1 **RSAT** is NP-complete.

Proof A restricted version of SAT, where each clause is the disjunction of no more than 3 literals, and conditions (4) holds, is NP-complete according to[10]. We show that conditions (1) (2) and (3) can be added one by one by polynomial time transformations such that each new (more restricted) problem is equivalent to the previous one.

To see that condition (1) can be added, we notice that if a clause C_j contains a single literal l, that literal must have value 1 if there is a satisfying assignment. If we remove C_j and all clauses containing l, and remove literal $\overline{l}$ from all clauses containing it, clearly, the reduced instance is satisfiable if and only if the original instance is satisfiable.

To see that condition (2) can be added, we notice that if a clause C_j contains both literals of a variable x_i, it is automatically satisfied since $x_i + \overline{x_i} = 1$. . Therefore, C_j can be removed from C without affecting the solution.

To see that condition (3) can be added, we notice that if a variable x_i has only one literal appearing in the clauses, we can assign value 1 to that literal (i.e. if x_i appears, assign $x_i = 1$; if $\overline{x_i}$ appears, assign $x_i = 0$), and remove the clauses containing it. Clearly, the reduced instance is satisfiable if and only if the original **SAT** instance is satisfiable.

All the three transformations reduce the number of clauses, so by repeatedly applying them on the original instance, in no more than m steps we will eventually get a reduced instance which meets all conditions (1 - 5), and which is satisfiable if and only if the original instance is satisfiable. The transformation is clearly in polynomial time. □

Note that conditions (3) and (4) indicate that each literal will appear in either one or two clauses. This is important to our construction of the transformations.

3.1. The General Mapping Problem

We first define the decision version of the general delay-optimal mapping problem under the nominal delay model.

Problem: Depth-Bounded LUT Mapping under Nominal Delay Model (**DBLMN**)
Instance: A constant $K \geq 2$, a Boolean network N of p nodes and q edges such that for any $v \in N$, $|input(v)| \leq K$, and three constants $d_L \geq 0$, $d_N \geq 1$, and B.
Question: Under the nominal delay model with parameters d_L, d_N, is there a K-LUT mapping solution of N that has delay no more than B?

We shall construct a polynomial-time transformation that transforms each instance of **RSAT** to an instance of **DBLMN**. Intuitively, we want to relate the decision of the truth assignment of a Boolean variable in an instance F of **RSAT** to the decision of a node duplication in the corresponding network N of the **DBLMN** instance. Since determining the truth assignment is difficult, we can show that determining the node duplication is also difficult.

Given an instance F of the **RSAT** with n variables $x_1, x_2, ..., x_n$ and m clauses $C_1, C_2, ..., C_m$, we construct a K-bounded Boolean network N corresponding to the instance F as follows.

First, for each variable x_i, we construct a subnetwork $N(x_i)$ that consists of the following nodes:

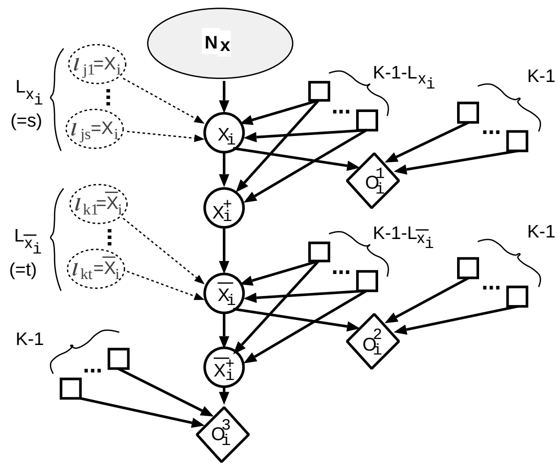

Figure 2 Transformation of **RSAT** instance to **DBLMN** instance:
Construction of $N(x_i)$ for variable x_i.

- Four internal nodes denoted as x_i, x_i^+, $\overline{x_i}$, and $\overline{x_i^+}$;
- $5(K-1)-(L_{x_i}+L_{\overline{x_i}})$ PI nodes, where L_{x_i} and $L_{\overline{x_i}}$ are the number of clauses containing x_i and $\overline{x_i}$ respectively;
- A *super-PI* node N_x, which is a subnetwork to be defined later;
- Three PO nodes denoted as O_i^1, O_i^2, O_i^3;

The nodes are connected as shown in Figure 2, where N_x, x_i, x_i^+, $\overline{x_i}$, $\overline{x_i^+}$, and O_i^3 form a direct chain, while x_i and $\overline{x_i}$ are connected to O_i^1 and O_i^2 respectively. There are $K-1$ PI nodes connected to each of the PO nodes, $K-1-L_{x_i}$ PI nodes connected to both x_i and x_i^+, and similarly $K-1-L_{\overline{x_i}}$ PI nodes connected to both $\overline{x_i}$ and $\overline{x_i^+}$. Note that for $K \geq 3$, $K-1-L_{x_i}$ and $K-1-L_{\overline{x_i}}$ will never be less than zero, since L_{x_i} and $L_{\overline{x_i}}$ are at most 2. Therefore, for $K \geq 3$ the construction is feasible. We call this subnetwork the subnetwork of x_i.

Next, for each clause C_j with 3 literals l_j^1, l_j^2, l_j^3, we construct a subnetwork $N(C_j)$ consisting of the following nodes:

- Three internal nodes denoted as l_j^1, l_j^2, and l_j^3;
- $3K-2$ PI nodes;
- One *super-PO* node N_c, which is a subnetwork to be defined later.

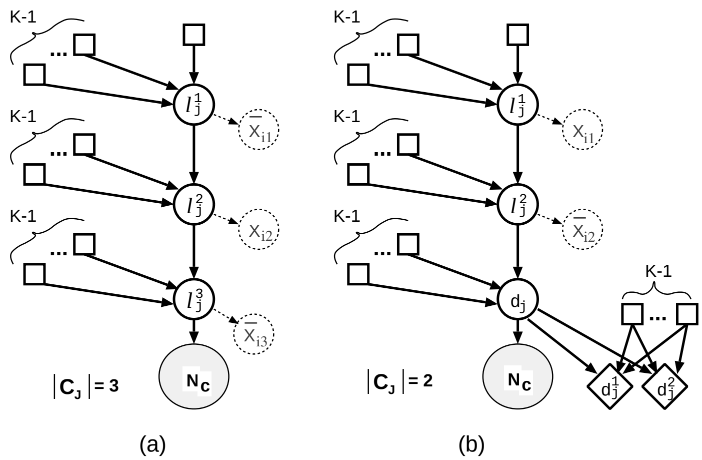

Figure 3 Transformation of **RSAT** instance to **DBLMN** instance:
Construction of $N(C_j)$ for clause C_j.

The connection is shown in Figure 3(a), where l_j^1, l_j^2, l_j^3 and N_c form a direct chain, and each has K, $K-1$, $K-1$, and 0 PI nodes connected to it respectively.

If the clause C_j contains only 2 literals l_j^1 and l_j^2, the subnetwork $N(C_j)$ is constructed with the following nodes:

- Two internal nodes l_j^1, l_j^2, and a dummy internal node d_j;
- $4K-3$ PI nodes;
- two PO nodes d_j^1 and d_j^2;
- One super-PO node N_c.

The connection as shown in Figure 3(b) is similar to the case of 3-literal clause, except that the

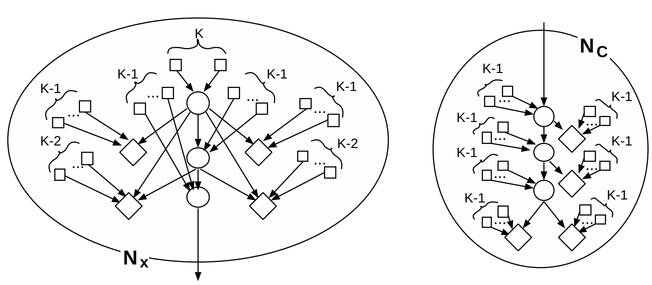

Figure 4 Transformation of **RSAT** instance to **DBLMN** instance:
Construction of *super nodes* N_x and N_C.

additional $K-1$ PI nodes and two PO nodes are attached to d_j.

The super-PI node N_x and the super-PO node N_c are shown in Figure 4. N_x is a three level subnetwork consisting of 3 internal nodes, 4 PO nodes, and $7K-8$ PI nodes, and has one outgoing edge. N_c is a four level subnetwork with 3 internal nodes, 4 PO nodes, and $7K-7$ PI nodes, and has one incoming edge. The subnetworks N_x and N_c are used to balance the depth of different paths in N, which is necessary to the proof.

Finally, we connect the subnetworks $N(C_j)$, $j=1, 2, ..., m$, with the subnetworks $N(x_i)$, $i=1, 2, ..., n$. Let l_j^r ($1 \le r \le 3$) be a literal in C_j. If for some variable x_i, $l_j^r = x_i$, we connect the internal node l_j^r of subnetwork $N(C_j)$ to the internal node x_i of subnetwork $N(x_i)$. Similarly, if $l_j^r = \overline{x_i}$, we connect node l_j^r of $N(C_j)$ to node $\overline{x_i}$ of $N(x_i)$. We call x_i (or $\overline{x_i}$) the *variable node* of l_j^r, and call l_j^r a *literal node* of x_i (or $\overline{x_i}$). Such connections are illustrated in Figure 5, and also in Figures 2 and 3 (in dashed lines).

Figure 6 shows an example of an **RSAT** instance and the corresponding network for **DBLMN**.

It is clear that the transformation defined above takes $O(K(m+n))$ time.

Regarding the network N obtained through this transformation, we can show that N is K-bounded, and the only ways of packing more than one node into one K-LUT are to pack x_i into the K-LUT of x_i^+, and to pack $\overline{x_i}$ into the K-LUT of $\overline{x_i^+}$, for $1 \le i \le n$. Moreover, every node must be implemented by a K-LUT regardless such packing (i.e. if such a packing happens, node

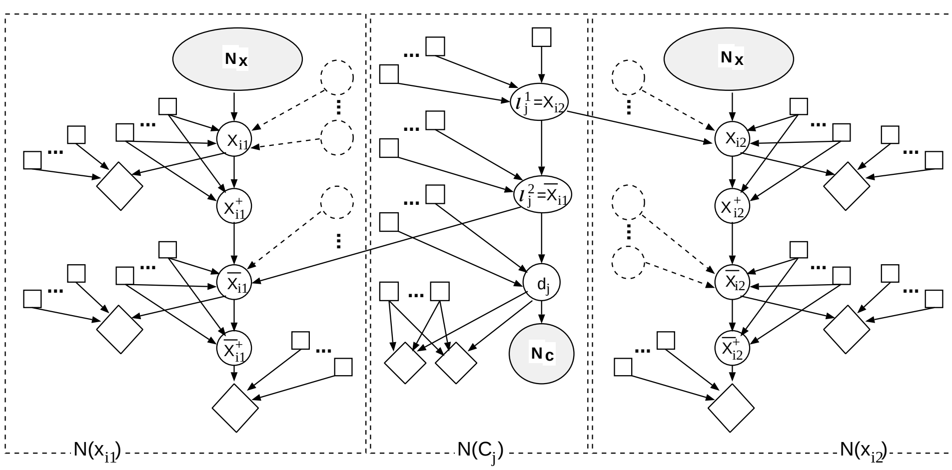

Figure 5 Transformation of **RSAT** instance to **DBLMN** instance:
Connections between $N(x_i)$ and $N(C_j)$.

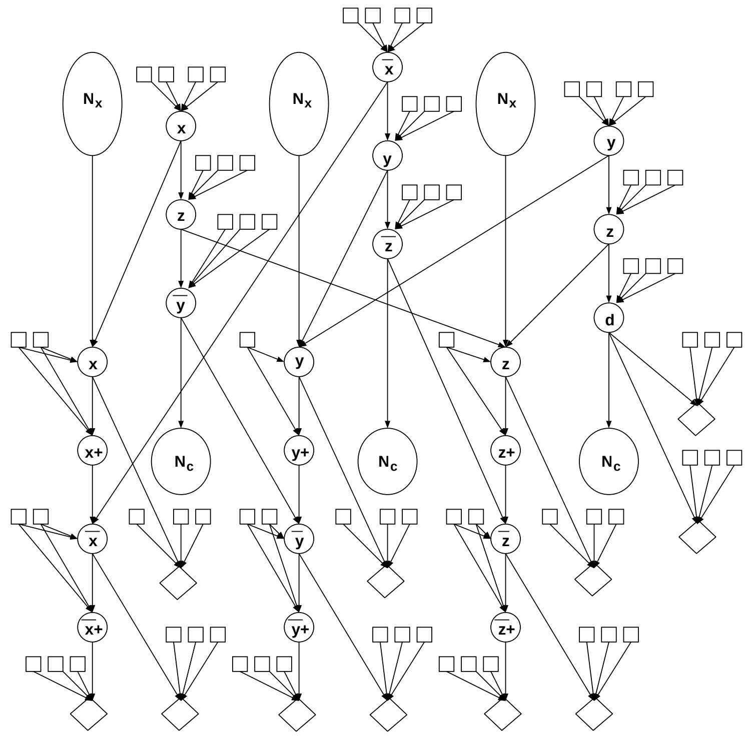

Figure 6 Example of transformation when $F = (x + \overline{y} + z)(\overline{x} + y + \overline{z})(y + z)$ and K=4.

duplication must happen). By linking the variable assignment of $x_i = 0$ for the **RSAT** instance F with the duplication and packing of node x_i in the network N (and assignment of $x_i = 1$ with the duplication and packing of $\overline{x_i}$), we can show the NP-completeness of the **DBLMN** problem. We start with a lemma about the transformation from a satisfiable **RSAT** instance to a delay-bounded **DBLMN** instance.

Lemma 2 If F is satisfiable, then N has a mapping solution of delay $7d_L + 15d_N$.

Proof Let $v(x_i)$ be the value of x_i in the truth assignment that satisfies F. We compute the mapping solution S as follows. For $i = 1, 2, ..., n$, if $v(x_i) = 0$, we perform $pack(x_i)$ after duplicating node x_i. Similarly, if $v(x_i) = 1$, we duplicate node $\overline{x_i}$, and perform $pack(\overline{x_i})$. The other copy of a duplicated node is implemented by a separate K-LUT, as is every other non-PI node in N.

Clearly, in each subnetwork $N(x_i)$, exactly one of $pack(x_i)$ and $pack(\overline{x_i})$ is performed, and one of the nodes x_i and $\overline{x_i}$ is duplicated. On the other hand, because the assignment satisfies F, each clause C_j contains at least one literal l_j^r with value 1, corresponding to an internal node in subnetwork $N(C_j)$ whose variable node in some subnetwork $N(x_i)$ is *not* duplicated.

We now compute the delay of the mapping solution. There are three types of paths from the PI nodes to the PO nodes in N:

(1) Paths entirely inside $N(x_i)$. It is not hard to see that the longest one is from N_x to O_i^3, regardless the possible packing. Denote the maximum delay along such a path as D_x. In our mapping solution, exactly one of the two variable nodes is duplicated and then packed along the path. This will eliminate one node level and one visible net (which is previously of size 2), but will increase the size of each input net of the duplicated node by one (see the mapping example in Figure 7). The overall effect of the packing is the delay reduction of $d_L + d_N$. The total delay along the path is

$$D_x = (\ 3d_L + 9d_N\ ) + 4d_L + (2{+}1{+}2{+}1)d_N = 7d_L + 15d_N. \qquad \text{(d)}$$

Therefore, this type of paths have delays no larger than $7d_L + 15d_N$.

(2) Paths entirely inside $N(C_j)$. Clearly, the longest one must pass through all the internal nodes and N_C. Denote the maximum delay along such a path as D_C. In our mapping solution, there exists at least one literal node whose variable node is not duplicated, so at least one of the literal nodes has fanout size 2, while the other may have fanout sizes 3, increased from 2 by the duplication of their variable nodes. Therefore, the overall effect of the duplication is the delay increase of up to $2d_N$ (see example in Figure 7). The total delay along the path is then

$$D_C \le 3d_L + (1{+}2{+}3{+}3)d_N + (\ 4d_L + 6d_N\ ) = 7d_L + 15d_N. \qquad \text{(e)}$$

Therefore, this type of paths also have delays bounded by $7d_L + 15d_N$.

(3) Paths from $N(C_j)$ to $N(x_i)$. The longest path of this type will go through all the internal nodes of $N(C_j)$ and all the internal nodes of $N(x_i)$, i.e. when $l_j^3 = x_i$. Denote the maximum delay along such a path as D_{Cx}. In our mapping solution, the total delay can be derived based on the discussions in (1) and (2):

$$D_{Cx} \le (\ 3d_L + 9d_N\ ) + (\ 4d_L + 6d_N\ ) = 7d_L + 15d_N. \qquad \text{(f)}$$

Therefore, this type of paths again have delay no larger than $7d_L + 15d_N$.

In summary, our mapping solution S of N, constructed based on the truth assignment that satisfies F, has maximum delay of $7d_L + 15d_N$. □

In order to derive a truth assignment that satisfies F from any delay-bounded mapping solution of N , we first analyze the characteristics of such a mapping solution.

Lemma 3 In a mapping solution of N with delay bound $7d_L + 15d_N$, at least one of the operations $pack(x_i)$ and $pack(\overline{x_i})$ must performed for each i, $1 \le i \le n$.

Proof If for some i, neither $pack(x_i)$ nor $pack(\overline{x_i})$ is performed, the longest path in $N(x_i)$ will have delay

$$D_x = (\ 3d_L + 9d_N\ ) + 5d_L + (1{+}2{+}1{+}2{+}1)d_N = 8d_L + 16d_N > 7d_L + 15d_N. \qquad (g)$$

Therefore, at least one of the two operations must be performed. □

Lemma 4 In a mapping solution of N with delay bound $7d_L + 15d_N$, for each $N(C_j)$, $1 \le j \le m$, at least one of the operations $pack(l_j^1)$, $pack(l_j^2)$, and (when $|C_j| = 3$) $pack(l_j^3)$ *cannot* be performed.

Proof If for some clause C_j, all the above operations are performed, the longest path in $N(C_j)$ will have delay

$$D_C = 3d_L + (1{+}3{+}3{+}3)d_N + (\ 4d_L + 6d_N\ ) = 7d_L + 16d_N > 7d_L + 15d_N. \qquad (h)$$

Therefore, at least one of these operations must not be performed. □

According to Lemmas 3 and 4, if a mapping solutions satisfies delay bound $7d_L + 15d_N$, it will be similar to the one we constructed in the proof of Lemma 2, except that for some x_i, it is possible that both $pack(x_i)$ and $pack(\overline{x_i})$ are performed. In such a case, however, the largest delay of a path in $N(x_i)$ is smaller than $7d_L + 15d_N$, and one of the packing operations is not necessary, hence can be ignored in the construction of a truth assignment from the mapping solution.

Lemma 5 If N has a mapping solution S of delay no more than $7d_L + 15d_N$, then F is satisfiable.

Proof The truth assignment for F is constructed as follows. For each variable x_i, if $pack(\overline{x_i})$ is performed in the mapping, we assign $v(x_i) = 1$; otherwise we assign $v(x_i) = 0$. That is, we ignore $pack(x_i)$ if $pack(\overline{x_i})$ is performed.

For any clause C_j with literals l_j^1, l_j^2, and (if $|c_j| = 3$) l_j^3, according to Lemma 4, at least one operation $pack(l_j^r)$ for some $1 \le r \le 3$ is not performed. If $l_j^r = x_i$, then $pack(x_i)$ is not performed. According to Lemma 3, this implies that $pack(\overline{x_i})$ is performed, and consequently we have assigned $v(x_i) = 1$. Therefore, C_j is satisfied. If $l_j^r = \overline{x_i}$, We know that $pack(\overline{x_i})$ is not performed, and consequently we have assigned $v(x_i) = 0$, i.e. $v(\overline{x_i}) = 1$, so C_j is also satisfied. Therefore, every clause of F is satisfied by the assignment. □

Therefore, by setting $B = 7d_L + 15d_N$, we can show

Theorem 1 **DBLMN** is NP-complete for $K \ge 3$.

Proof It is easily seen the the transformation takes polynomial time. Moreover, **DBLMN** is in NP because given any K-LUT mapping solution, we can easily verify if its delay is bounded by

B. Finally, according to Lemmas 2 and 5, the **RSAT** instance *F* has a *yes* answer if and only if the **DBLMN** instance *N* has a *yes* answer. Therefore, the NP-completeness of **RSAT** implies that **DBLMN** is NP-complete. □

Figure 7 illustrates the correspondence using the example in Figure 6. The duplicated part is drawn in heavy line.

Based on Theorem 1, we have

Corollary 1 The depth-optimal K-LUT mapping problem under nominal delay model is NP-hard when $K \geq 3$. □

Note that the construction of *N* does not apply when $K=2$, so the complexity of the problem is still open for $K=2$.

3.2. The Duplication-Free Mapping & Tree-Based Mapping Problems

In the preceding subsection, we have proved the NP-hardness of the general LUT mapping problem under nominal delay model by relating the truth assignment of Boolean variables in an

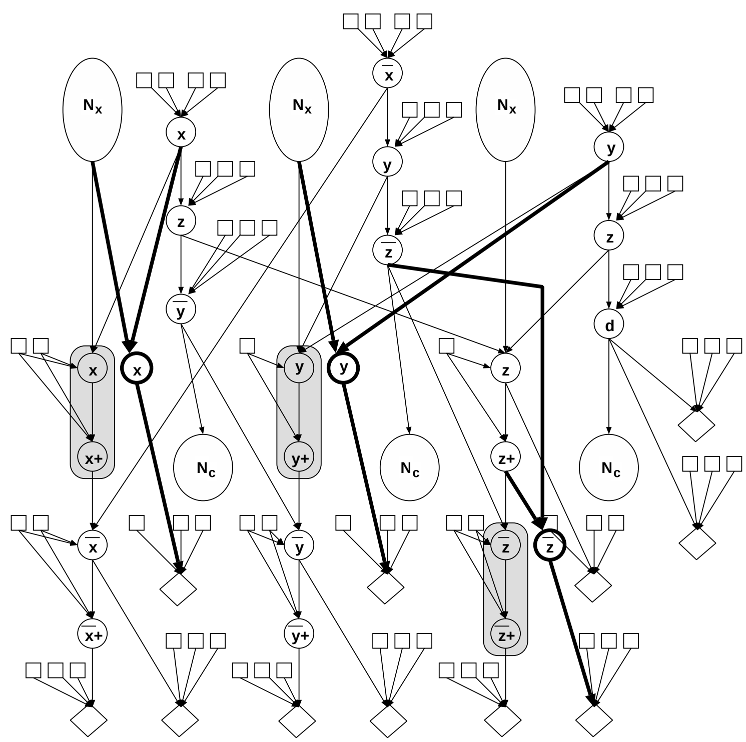

Figure 7 Depth-optimal mapping for the example in Figure 6 (K=4).
(Corresponding assignment for $F=(x+\bar{y}+z)(\bar{x}+y+\bar{z})(y+z)$: $x=y=0$, $z=1$)

RSAT instance to the duplication of nodes in the mapping of corresponding network. An interesting question is whether node duplication is the only cause of the difficulty. In this subsection we shall show that even when node duplication is prohibited, LUT mapping under nominal delay is still NP-hard when $K \geq 5$. This result indicates that the delay minimization problem is more difficult than the area minimization problem, which is polynomial-time solvable for duplication-free mapping. Our proof will also apply to the tree-based mapping problem, which is another indication that nominal delay minimization in LUT mapping is extremely difficult.

We first define the decision version of the duplication-free mapping problem.

Problem: Duplication-Free Depth-Bounded LUT Mapping under Nominal Delay Model (**DFDBLMN**)
Instance: A constant $K \geq 2$, a Boolean network N of n nodes and m edges such that for any $v \in N$, $|input(v)| \leq K$, and three constants $d_L \geq 0$, $d_N \geq 1$, and B.
Question: Under the nominal delay model with parameters d_L, d_N, is there a K-LUT mapping solution S of N such that every node belongs to only one K-LUT, and S has delay no more than B?

Again, we construct a polynomial-time transformation from an instance of **RSAT** to an instance of **DFDBLMN**. Since node duplication is not allowed, we link the Boolean variable assignment to the possible packing of multiple fanouts of the same node into a single K-LUT, and show that even the choice of packing operations only is difficult.

Given an instance F' of **RSAT** with n variables x_1, ..., x_n and m clauses C_1, ..., C_m. We construct a K-bounded Boolean network N' as follows.

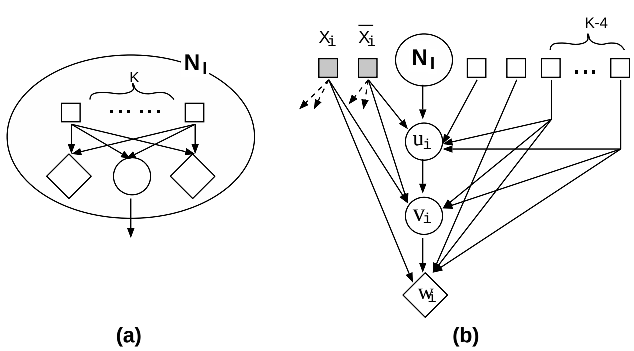

Figure 8 Transformation of **RSAT** instance to **DFDBLMN** instance:
Construction of $N'(x_i)$ for variable x_i.

For each variable x_i, we construct a subnetwork $N'(x_i)$ of $2K$ PIs, 3 internal nodes, and 3 POs as shown in Figure 8. Among them, K PIs, one internal node and 2 POs are used to form a *super-PI* named N_I shown in Figure 8(a). The other nodes are connected as in Figure 8(b), in which two of the PIs are used to denote the two literals of the variable, x_i and $\overline{x_i}$.

For each clause C_j of three literals, we construct a subnetwork $N'(C_j)$ of $2K$ PIs, 4 internal nodes, and one PO, as shown in Figure 9(a). Three of the PI nodes are used to denote the three literals in the clause, l_j^q $(1 \le q \le 3)$. If C_j has two literals, the third PI node (which otherwise denotes the third literal), will be connected to 3 additional POs. Moreover, $K-1$ additional PIs are also added to connect to these three POs. This case is illustrated in Figure 9(b).

The subnetworks are connected by merging the PI nodes that denote the same literals. Moreover, if any of these specially PI nodes has fewer than 4 fanouts, a dummy PO node is introduced as its fanout. (Note that because of the conditions (3)(4) of **RSAT**, each of these PIs has either 3 or 4 fanouts.) An example is shown in Figure 10, which is the network N' corresponding to the **RSAT** instance $F' = (x + y + \overline{z})(x + \overline{y} + z)(\overline{x} + y + z)(\overline{x} + \overline{y})$.

It is easily seen that for $K \ge 5$, the construction is feasible and takes polynomial time, and the network is K-bounded. Moreover, the only cases where two or more nodes can be packed into one K-LUT are (1) to pack u_i with v_i, or v_i with w_i, but not both, in Figure 8(b); (2) to pack one or two of r_j^q, $1 \le q \le 3$, but not all three, with s_j in Figure 9. Note that the packing of u_i with v_i or v_i with w_i will decrease the fanout size of the PI node $\overline{x_i}$ or x_i respectively. Finally, it can be seen that there is no non-PI nodes in the network N' that have multiple fanout. In other words, N' is *internal fanout-free*. Therefore, any mapping solution will be duplication-free.

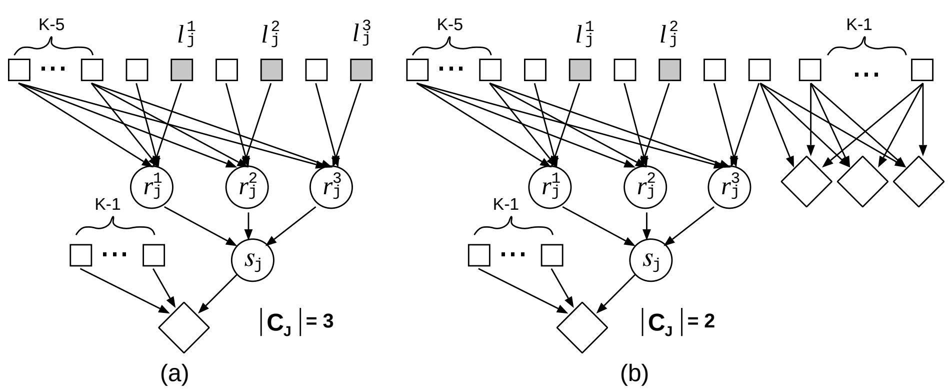

Figure 9 Transformation of **RSAT** instance to **DFDBLMN** instance: Construction of $N'(C_j)$ for clause C_j.

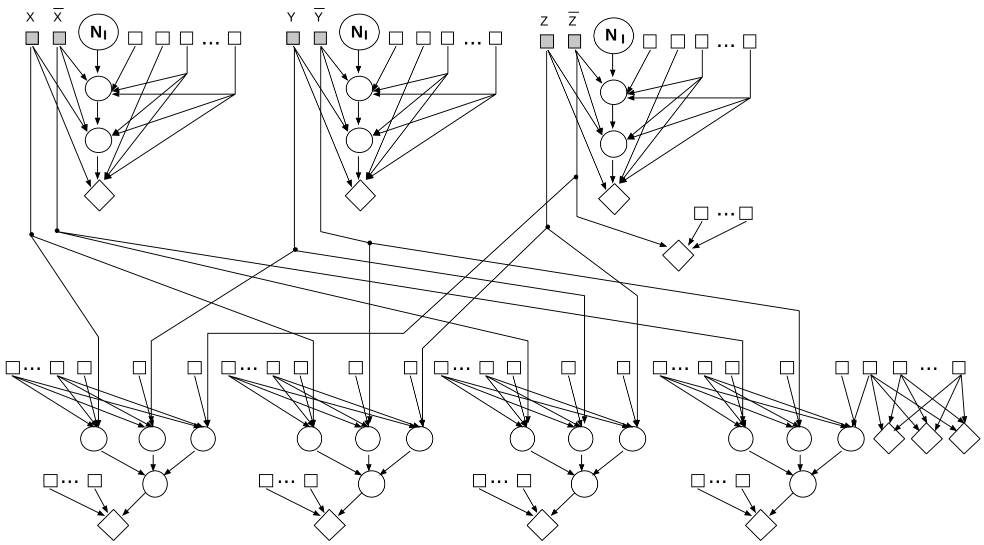

Figure 10 Example of transformation (and connection):
$F' = (x+y+\overline{z})(x+\overline{y}+z)(\overline{x}+y+z)(\overline{x}+\overline{y})$.

By linking the variable assignment $x_i = 1$ for F' with the packing of v_i with w_i in N' (and the assignment $x_i = 0$ with the packing u_i with v_i), we can show

Lemma 6 If F' is satisfiable, then N' has a mapping solution S of delay no more than $3d_L + 5d_N$.

Proof We construct the mapping solution of N' as follows. For each variable x_i, if $v(x_i) = 1$, we pack v_i with w_i into the same K-LUT in $N'(x_i)$, while if $v(x_i) = 0$, we pack u_i with v_i into the same K-LUT. Moreover, in $N'(C_j)$ we pack r_j^q with s_j if and only if l_j^q ($1 \le q \le 3$) is assigned value 0. All other non-PI nodes are implemented separately.

The maximum delay in $N'(x_i)$, regardless which packing is performed, will be

$$D'_x = 3d_L + (3+1+1)d_N = 3d_L + 5d_N, \tag{i}$$

which is the delay from N_I to w_i. The maximum delay in $N'(C_j)$ is determined by the path from a PI node l_j^q corresponding to a literal of value 0 to the PO node, going through two internal nodes of single fanout. Due to the packing in $N'(x_i)$, the fanout of l_j^q is 3. Therefore, the maximum delay in $N'(C_j)$ is

$$D'_C = 3d_L + (3+1+1)d_N = 3d_L + 5d_N, \tag{j}$$

the same as D'_x. Therefore, the delay of the mapping solution is bounded by $3d_L + 5d_N$. □

Lemma 7 If N' has a mapping solution with delay no more than $3d_L + 5d_N$, then F' is satisfiable.

Proof We determine the truth assignment for F' as follows. If in $N'(x_i)$ v_i is packed with w_i into the same K-LUT, we assign $v(x_i) = 1$; otherwise we assign $v(x_i) = 0$. Note that if v_i is *not* packed with w_i, we must have u_i packed with v_i, because if none of the packing is performed, the delay of $N'(x_i)$ would be larger than $3d_L + 5d_N$. Equivalently, if we assign $v(x_i) = 1$, the PI node x_i will have its fanout size reduced from 4 to 3, and the PI node $\overline{x_i}$ will have its fanout remain 4; while if we assign $v(x_i) = 0$, the PI node $\overline{x_i}$ will have its fanout size reduced from 4 to 3, and the PI node x_i will have its fanout remain 4.

Now for any clause C_j, if all of the literals l_j^q, $1 \le q \le 3$ (or $1 \le q \le 2$ if C_j has only two literals), are assigned value 0, then, all the corresponding PI nodes l_j^q will have fanout size of 4. On the other hand, not all of the nodes r_j^q, $1 \le q \le 3$, can be packed with s_j in the same time, and no matter which one is *not* packed, the node will have a fanin node with fanout size 4, and the delay from that node to the PO node will be

$$D'_C = 3d_L + (4{+}1{+}1)d_N = 3d_L + 6d_N > 3d_L + 5d_N, \tag{k}$$

which is a contradiction. Therefore, at least one of the literals in C_j must have been assigned 1, so C_j is satisfied. □

Therefore, by setting $B = 3d_L + 5d_N$, we can prove

Theorem 2 **DFDBLMN** is NP-complete for $K \ge 5$.

Proof It is easily seen that **DFDBLMN** is in NP and the transformation is polynomial-time. From Lemmas 1, 6 and 7, we know that **DFDBLMN** is NP-complete. □

Corollary 2 The depth-optimal duplication-free K-LUT mapping problem under nominal delay model is NP-hard when $K \ge 5$. □

Moreover, since the network N' constructed above is internal fanout free, the tree-decomposition will produce a single tree for each of the $N'(x_i)$ and $N'(C_j)$ subnetworks. Therefore, the tree-based mapping on N' is equivalent to the duplication-free mapping. Hence we have

Corollary 3 The depth-optimal tree-based K-LUT mapping problem under nominal delay model is NP-hard when $K \ge 5$. □

Note that the construction of N' is feasible only when $K \ge 5$. For $K < 5$, the complexity problem is open except for K=2, which can be solved in polynomial time in the next section.

4. Mapping Algorithms for Nominal Delay Minimization

In this section we first present a polynomial time optimal algorithm for a special case ($K=2$) of duplication-free mapping under the nominal delay model. This is mostly of theoretical interest, since K is usually larger than 2 in practice, and delay-optimized mapping solutions tend to have node duplications. Then, for the general case of $K \geq 2$, we present a simple heuristic to demonstrate the benefit of using nominal delay model in delay minimization.

4.1. Optimal Duplication-Free Mapping for $K = 2$

As defined in [5], the *maximum fanout-free cone* (MFFC) of a node v, denoted $MFFC_v$, is an FFC of v such that for any node w, $w \in MFFC_v$ if $output(w) \subseteq MFFC_v$. One property of MFFC is that a network can be decomposed into a set of disjoint MFFCs. The MFFC decomposition is important because every duplication-free mapping solution induces a mapping solution of each of the MFFCs in the decomposition. In the case of area minimization, this property allows us to map each MFFC independently and compose an optimal solution of the entire network from the optimal solutions of the MFFCs. In the case of nominal delay minimization, in general this method no longer works because the delay of some input nets to an MFFC can be changed due to the mapping of other MFFCs that share the same inputs net.

However, if $K=2$, we can still combine the optimal mapping solutions of the MFFCs to obtain an optimal mapping solution for the entire network. The optimal mapping solution in an MFFC is very easily computed for $K=2$, since any K-feasible cut is a nontrivial min-cut[3], and the following result is proved in [4].

Lemma 8 For any cone C_v of a node v, there exists a unique maximum-volume[4] min-cut $(X, \bar{X})$ of C_v such that for any other min-cut $(X', \bar{X'})$ of C_v, $\bar{X'} \subset \bar{X}$. □

According to Lemma 8, if v is the root of an MFFC $MFFC_v$, then, there is a unique 2-LUT LUT_v rooted at v defined by the maximum volume min-cut, such that for any other 2-LUT implementation LUT'_v in a duplication-free mapping solution, we have $LUT'_v \subset LUT_v$. Based on this property and the structure of the MFFC decomposition of $MFFC_v - LUT_v$, we can show that no matter how the other nodes in the network are mapped, the mapping solution for v using LUT_v always has the minimum delay. Therefore, the following algorithm computes a mapping solution with the minimum nominal delay:

---

[3] A trivial min-cut is a cut of size one, which can be identified easily if exists.

[4] The *volume* of a cut $(X, \bar{X})$ is defined to be $|\bar{X}|$.

1. Decompose the network into disjoint MFFCs;
2. For each MFFC $MFFC_v$, do
   2.1. Find the maximum-volume min-cut $(X, \overline{X})$ of cut-size 2, and assign $LUTv := \overline{X}$;
   2.2. If $MFFC_v - LUT_v \neq \emptyset$, map $MFFC_v - LUT_v$ recursively using this algorithm.

The cut computation in step 2.1 can be implemented using the augmenting path algorithm as used in FlowMap [4], which takes $O(m_v + n_v)$ time, where m_v and n_v are the number of edges and nodes in $MFFC_v$. If the network has m edges and n nodes, in the worst case we may need to compute the mapping for $\Omega(n)$ nodes, therefore, this algorithm takes no more than $O((m+n)n) = O(mn)$ time. That gives

Theorem 3 For $K = 2$, the delay-optimal duplication-free mapping problem under nominal delay model can be solved in $O(mn)$ time, where m and n are the number of edges and nodes in the network. □

Since Lemma 8 does not hold for K-feasible cuts when $K > 2$, this algorithm is not applicable when $K > 2$.

4.2. A Simple Heuristic for LUT Mapping under Nominal Delay Model

In this section, we present a heuristic algorithm for solving the general nominal delay minimization problem in LUT mapping for $K \geq 2$. We use the dynamic programming approach similar to that used in FlowMap[4]. We compute the delay-minimal mapping of each node according to a topological ordering of nodes starting from the PIs. The computation of delay-minimal mapping LUT_v of node v depends on the the delay-minimal mapping of predecessors of v computed in the previous steps. After we have computed the delay-minimal mapping LUT_v of v, we also record the delay to the output of LUT_v (excluding the fanout delay of LUT_v) in the mapping solution, and denote it as $l(u)$.

As we pointed out in Section 2, however, the mapping of node v may depend on the mapping of some nodes outside of fanin cone N_v of v, which may have not been processed (since they may appear later in the topological ordering). This is due to the fact that the fanout net delay of the nodes in N_v may not be known. In order to solve this problem, we propose a way to estimate the fanout net delays of the nodes in N_v.

For each node u in the fanin cone N_v of node v, we estimate the nominal delay $D_N(u)$ of the fanout net of u as follows:

$$D_N(u) = d_N \cdot \mid output(u) \mid + \alpha \cdot (\mid \bigcup_{w \in output(u)} output(w) \mid - \mid output(u) \mid), \tag{2}$$

where α is a positive constant for adjusting the relative weight of the second term. The first term depends on the fanout size of u. If it is large, it is likely that LUT_u will also have large fanout size, thus, large nominal delay. The second term is a correction of the simple estimation by the first term. In the second term, $\mid \bigcup_{w \in output(u)} output(w) \mid$ is the total fanout size of the fanouts of u. If it is smaller than $\mid output(u) \mid$, the fanouts of fanouts of u converges (see Figure 11(a)), thus some fanouts of u may be packed together in one or more LUTs. Thus, the nominal delay of LUT_u will decrease. On the other hand, if $\mid \bigcup_{w \in output(u)} output(w) \mid$ is larger, the fanouts of fanouts of u diverge (see Figure 11(b)). In this case, some fanouts of u are likely to be duplicated. Thus, the nominal delay of LUT_u will increase. Note that the value of $D_N(u)$ may be corrected later on when partial mapping solution is known.

Given the fanout delay estimation $D_N(u)$ of every node u in N_v and the delay to the fanout net of LUT_u in the delay-minimal mapping solution, we can compute the delay-minimal mapping of node v as follows. We compute a K-feasible cut $(X_v, \bar{X}_v)$ such that the *height* of this cut, defined as

$$H(X_v, \bar{X}_v) = \max\{l(u) + D_N(u) \mid u \in X_v\}, \tag{3}$$

is minimum. Note that $H(X_v, \bar{X}_v)$ corresponds to the maximum delay to the input of $\bar{X}_v$ (which will be the LUT implementing v). Such a cut can be computed using the minimum-height K-feasible cut algorithm introduced in FlowMap [4]. The minimum-height K-feasible cut algorithm was originally used in FlowMap for depth-optimal mapping under the unit delay model. The minimum-height K-feasible cut at each node can be found in $O(Km)$ time, where m is the number of edges in the network. Later on, the algorithm was generalized to delay-minimum mapping

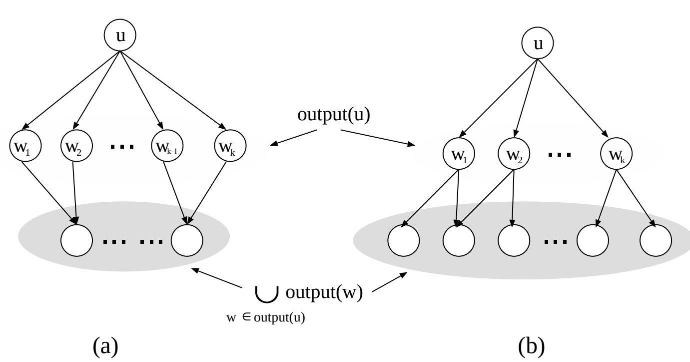

Figure 11 Nominal delay estimation.

when each net has a fixed pre-assigned delay of arbitrary value [3]. The generalized minimum-height K-feasible cut algorithm runs in $O(Km\log n)$ time, where m and n are the number of edges and nodes in the network, respectively. We use the generalized minimum-height K-feasible cut algorithm in our mapping of node v and use $D_N(u)$ as the fanout delay estimation of each node in N_v. After we find the minimum-height K-feasible cut $(X_v, \bar{X}_v)$ in N_v, we set $LUT_v = \bar{X}_v$. For some node u in N_v, it may have two or more fanouts packed into a single input to LUT_v. Therefore, we update the fanout net delay estimation of every node in $input(LUT_v)$. Finally, we compute the delay to the output of LUT_v in the mapping solution and record it in $l(v)$.

After we have computed delay-minimal mapping of each node, we shall generate all the necessary LUTs starting from the POs using the same procedure as in FlowMap, which can be done in $O(m)$ time. Since we have to compute a minimum-height K-feasible cut at each node, the time complexity of our algorithm is $O(Kmn\log n)$. Since the time complexity is bounded by a low order polynomial, this algorithm can handle circuits of large sizes very efficiently, and can be invoked several times for the same design with different choices of α and β in $D_N(u)$ estimation to obtain the best result.. The experiments that will be presented in the next section show that our heuristic algorithm gives consistent improvement over the FlowMap algorithm in terms of the actual delay of the final FPGA designs.

5. Experimental Results

We have implemented the heuristic algorithm described in the preceding section and tested it on a set of six mid-size MCNC benchmark circuits which were also used in [4]. These circuits were chosen because each of them can be mapped into a single Xilinx XC3000 series FPGA chip, and be placed and routed using Xilinx design tools, so that we were able to measure the real delays of the mapping solutions (the second entry in Table 3 shows the type of Xilinx XC3000 chip used for each example). The circuits have been optimized for delay minimization using technology independent synthesis and decomposed into two-input simple gate networks. Each circuit was first mapped using the heuristic algorithm into 5-LUTs. Then, two postprocessing operations for area minimization were applied. They were the *predecessor packing* and the *gate decomposition*, which were also used by FlowMap [4]. We restricted the operations such that no node duplication was allowed, so that the nominal delay will not increase. Then, we packed the LUTs into Xilinx XC3000 series 2-output CLBs, whenever possible, using a maximum matching algorithm[5]. Finally, we used the Xilinx **apr** program to place and route the mapping solutions, and used the Xilinx **xdelay** program to measure the actual delays. We selected the smallest

[5] Each XC3000 CLB can implement one 5-LUT, or two 4-LUTs with no more than 5 inputs in total [21].

possible FPGA chips that can accommodate the circuits and complete the routing. (Other mid-size benchmark circuits from the MCNC benchmark set were also mapped, but were not reported in Table 3 since they cannot fit into any XC3000 chip due to their large number of I/O pins, or incomplete routing.)

For comparison, we also mapped the circuits using FlowMap. The experimental procedure was the same except for the use of the mapping algorithm. The same size FPGA chips were used for placement/routing of the FlowMap mapping solutions, although FlowMap often uses fewer CLBs and sometimes can fit into smaller chips.

We set $d_N = 0.1 d_L$ in Eq.(2) in our nominal delay formulation [6]. For each circuit we tried 10 different α values ($\alpha = 0.1 d_N$, ..., $1.0 d_N$), and kept the one that produced the solution with the smallest nominal delay for placement and routing. Table 3 summarized these results. As one can see, with proper choice of parameters in the delay estimation model, our simple heuristic based on the nominal delay model can produce better mapping solutions than the depth-optimal mapping algorithm.

6. Conclusion

In this report, we have studied the LUT-based FPGA technology mapping problem for delay minimization under the nominal delay model. The nominal delay model is an improvement over the widely used unit delay model in terms of estimating net delays, yet is simple enough to be measured and considered during the mapping stage. Contrary to the fact that the depth minimization problem in LUT mapping is polynomial time solvable, we have shown that the nominal delay minimization problem is NP-hard for general LUT mapping when $K \geq 3$, and

circuit name	XC3000 part#	FlowMap			nominal delay heuristic			
		#clb	ndly	adly (*ns*)	α	#clb	ndly	adly (*ns*)
9sym	3020PC68	50	8.8	96.3	0.06	50	8.7	94.3
C880	3090PQ208	166	15.7	209.1	0.06	195	14.3	208.3
alu2	3064PC84	120	17.3	204.3	0.03	149	17.1	200.1
apex7	3042PP132	66	8.7	99.4	0.02	65	6.3	93.2
count	3020PC68	59	7.5	83.5	0.02	60	7.1	79.8
vg2	3020PC68	34	5.9	83.0	0.03	39	5.1	78.1

Table 3 Experimental results: ndly = nominal delay, adly = actual delay after routing (for nominal delay model, $d_L = 1$, $d_N = 0.1$)

[6] In fact, we tried $d_N = 0.5 d_L$, $0.2 d_L$, and $0.1 d_L$, and observed that $d_N = 0.1 d_L$ gives the best nominal delay estimation.

remains NP-hard for duplication-free mapping and tree-based mapping when $K \geq 5$. Also, we have presented an optimal algorithm for a special case ($K=2$) of nominal delay minimization in duplication-free mapping, as well as a heuristic algorithm that works for arbitrary K to minimize the nominal delay during general LUT mapping. We implemented and tested the algorithm on a set of MCNC benchmark circuits, and obtained designs of smaller delay than those produced by the depth-optimal mapping algorithm under the unit delay model.

Accurate delay modeling is very important at every stage of FPGA design automation. We have shown that the nominal delay model is a better approximation of the real net delay than the unit delay model. Currently, we are working on more effective heuristics for nominal delay minimization, as well as more accurate delay modeling based on the interaction between technology mapping and placement/routing steps.

References

[1] Bhat, N. and D. Hill, ‘‘Routable Technology Mapping for FPGAs,’’ *First Int’l ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 143-148, Feb. 1992.

[2] Chen, K. C., J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, ‘‘DAG-Map: Graph-based FPGA Technology Mapping for Delay Optimization,’’ *IEEE Design and Test of Computers*, pp. 7-20, Sep. 1992.

[3] Cong, J., Y. Ding, T. Gao, and K. Chen, ‘‘An Optimal Performance-Driven Technology Mapping Algorithm for LUT based FPGAs under Arbitrary Net-Delay Models,’’ *Proc. 1993 Int’l Conf. on CAD and Computer Graphics*, pp. 599-603, Aug. 1993.

[4] Cong, J. and Y. Ding, ‘‘An Optimal Technology Mapping Algorithm fo Delay Optimization in Lookup-Table Based FPGA Designs,’’ *Proc. IEEE Int’l Conf. on Computer-Aided Design*, pp. 48-53, Nov. 1992.

[5] Cong, J. and Y. Ding, ‘‘On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping,’’ *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 213-218, June 1993.

[6] Cong, J. and Y. Ding, ‘‘Beyond the Combinatorial Limit in Depth Minimization for LUT-Based FPGA Designs,’’ *Proc. IEEE Int’l Conf. on Computer-Aided Design*, pp. 110-114, 1993.

[7] Farrahi, A. and M. Sarrafzadeh, ''On the Lookup-Table Minimization Problem for FPGA Technology Mapping,'' in *Tech. Report 93-AC-102*, , Department of EECS, Northwestern University (July 1993).

[8] Francis, R. J., J. Rose, and Z. Vranesic, ''Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs,'' *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 613-619, June 1991.

[9] Francis, R. J., J. Rose, and Z. Vranesic, ''Technology Mapping of Lookup Table-Based FPGAs for Performance,'' *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 568-571, Nov. 1991.

[10] Garey, M. and D. Johnson, *Computer and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco (1979).

[11] Hill, D., ''A CAD System for the Design of Field Programmable Gate Arrays,'' *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 187-192, June 1991.

[12] Karplus, K., ''Xmap: A Technology Mapper for Table-lookup Field-Programmable Gate Arrays,'' *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 240-243, June 1991.

[13] Murgai, R., Y. Nishizaki, N. Shenay, R. Brayton, and A. Sangiovanni-Vincentelli, ''Logic Synthesis Algorithms for Programmable Gate Arrays,'' *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 620-625, 1990.

[14] Murgai, R., N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, ''Performance Directed Synthesis for Table Look Up Programmable Gate Arrays,'' *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 572-575, Nov. 1991.

[15] Murgai, R., N. Shenoy, R. K. Brayton, and A. Sangiovanni-Vincentelli, ''Improved Logic Synthesis Algorithms for Table Look Up Architectures ,'' *Proc. IEEE Int'l Conf. on Computer-Aided Design*, pp. 564-567, Nov. 1991.

[16] Sawkar, P. and D. Thomas, ''Technology Mapping for Table-Look-Up Based Field Programmable Gate Arrays,'' *ACM/SIGDA Workshop on Field Programmable Gate Arrays*, pp. 83-88, Feb. 1992.

[17] Sawkar, P. and D. Thomas, ''Performance Directed Technology Mapping for Look-Up Table Based FPGAs,'' *Proc. 30th ACM/IEEE Design Automation Conf.*, pp. 208-212, June 1993.

[18] Schlag, M., P. Chan, and J. Kong, ''Empirical Evaluation of Multilevel Logic Minimization Tools for a Field Programmable Gate Array Technology,'' *Proc. 1st*

Int'l Workshop on Field Programmable Logic and Applications, Sept. 1991.

[19] Schlag, M., J. Kong, and P. K. Chan, ''Routability-Driven Technology Mapping for Lookup Table-Based FPGAs,'' *Proc. 1992 IEEE International Conference on Computer Design*, pp. 86-90, Oct. 1992.

[20] Woo, N.-S., ''A Heuristic Method for FPGA Technology Mapping Based on the Edge Visibility,'' *Proc. 28th ACM/IEEE Design Automation Conference*, pp. 248-251, June 1991.

[21] Xilinx, *The Programmable Gate Array Data Book*, Xilinx, San Jose (1992).